# MDL Assignment 2 Part 2 Report


### Team: SendBobstoAlice (108)

### Members:

- Naman Ahuja (2019101042)
- Nikhil Chandak (2019111040)
- Shashwat Goel (2019111006)


`STEP_COST = -20`

# Task 1

It took 124 iterations for the Value Iteration algorithm to converge. The policy has a mix of diverse actions. When MM is ready and IJ is in the safe positions (`N`, `S`, `W`), IJ prefers to `STAY` to not get hurt by MM. At `W`, IJ prefers to move right to `C` mostly and `SHOOT` (if he has arrows) to attack MM. At `N`, IJ prefers to move down to `C` mostly and `CRAFT` (if he has less than 3 arrows) so as to gain arrows. At `E`, IJ mostly tries to `SHOOT` and `HIT` depending upon the health of MM and the arrows IJ has. At `S`, IJ prefers to move up to `C` if MM is dormant.

At `C`, if MM is `READY` and IJ has arrows, then he mostly tries to `SHOOT`. If MM is dormant and IJ has arrows, he mostly moves to the right unless he has $\geq 2$ arrows and MM's health is $25$ when he tries to `SHOOT`. The fact that he has 2 arrows means there's a very high that in 2 actions IJ kills MM with health $25$. If he moved elsewhere, he would anyway need to spend at least 1 more action from the new position and the probability of killing MM would only be worse. If IJ has enough $(2)$ material and no arrows, he moves up to `N` . If IJ has only 1 material and no arrows, then he may move to `N` or `E` depending on the health of MM.

It is to be noted that IJ never gathers at `S` since it will be helpful only when IJ goes back to `N` to make arrows and then come back to `E` to hit MM. The fact that step cost is very high (negative), so it is better to reach `E` early so .

## Start State = (W, 0, 0, D, 100)

After finding the optimal policy, we simulate the policy from the given start state. As the transitions are probabilistic, so simulation will not be deterministic. So as examples, we show below 2 such instances—

Simulation Example 1:

```
START STATE: ('W', 0, 0, 'D', 100)
('W', 0, 0, 'D', 100) RIGHT ('C', 0, 0, 'D', 100) -20.0
('C', 0, 0, 'D', 100) RIGHT ('E', 0, 0, 'D', 100) -39.98
('E', 0, 0, 'D', 100) HIT ('E', 0, 0, 'D', 50) -59.94
('E', 0, 0, 'D', 50) HIT ('E', 0, 0, 'D', 0) -79.88
Overall Reward: -29.88
```

Simulation Example 2

```
START STATE: ('W', 0, 0, 'D', 100)
('W', 0, 0, 'D', 100) RIGHT ('C', 0, 0, 'D', 100) -20.0
('C', 0, 0, 'D', 100) RIGHT ('E', 0, 0, 'R', 100) -39.98
('E', 0, 0, 'R', 100) HIT ('E', 0, 0, 'R', 100) -59.94
('E', 0, 0, 'R', 100) HIT ('E', 0, 0, 'D', 100) -119.76
('E', 0, 0, 'D', 100) HIT ('E', 0, 0, 'D', 100) -139.68
('E', 0, 0, 'D', 100) HIT ('E', 0, 0, 'R', 100) -159.581
('E', 0, 0, 'R', 100) HIT ('E', 0, 0, 'R', 100) -179.461
('E', 0, 0, 'R', 100) HIT ('E', 0, 0, 'R', 50) -199.321
('E', 0, 0, 'R', 50) HIT ('E', 0, 0, 'R', 50) -219.162
('E', 0, 0, 'R', 50) HIT ('E', 0, 0, 'R', 0) -238.983
```

We find after multiple simulations that the best strategy for IJ is to directly head from `W` to `E` and then repeatedly `HIT` MM. The fact that we have no arrows and a high negative `STEP_COST` means we should try to end the game as soon as possible which is what IJ tries to do.

## Start State = (C, 2, 0, R, 100)

After finding the optimal policy, we simulate the policy from the given start state. As the transitions are probabilistic, so simulation will not be deterministic. So as examples, we show below 2 such instances —

Simulation Example 1:

```
START STATE: ('C', 2, 0, 'R', 100)
('C', 2, 0, 'R', 100) UP ('N', 2, 0, 'R', 100) -20.0
('N', 2, 0, 'R', 100) CRAFT ('N', 1, 1, 'R', 100) -39.98
('N', 1, 1, 'R', 100) CRAFT ('N', 0, 3, 'D', 100) -59.94
('N', 0, 3, 'D', 100) DOWN ('C', 0, 3, 'D', 100) -79.88
('C', 0, 3, 'D', 100) RIGHT ('E', 0, 3, 'D', 100) -99.8
('E', 0, 3, 'D', 100) SHOOT ('E', 0, 2, 'D', 75) -119.7
('E', 0, 2, 'D', 75) SHOOT ('E', 0, 1, 'D', 50) -139.581
('E', 0, 1, 'D', 50) SHOOT ('E', 0, 0, 'R', 50) -159.441
('E', 0, 0, 'R', 50) HIT ('E', 0, 0, 'D', 75) -218.963
('E', 0, 0, 'D', 75) HIT ('E', 0, 0, 'D', 75) -238.784
('E', 0, 0, 'D', 75) HIT ('E', 0, 0, 'D', 75) -258.584
('E', 0, 0, 'D', 75) HIT ('E', 0, 0, 'D', 75) -278.366
('E', 0, 0, 'D', 75) HIT ('E', 0, 0, 'D', 75) -298.127
('E', 0, 0, 'D', 75) HIT ('E', 0, 0, 'R', 25) -317.868
('E', 0, 0, 'R', 25) HIT ('E', 0, 0, 'R', 25) -337.59
('E', 0, 0, 'R', 25) HIT ('E', 0, 0, 'R', 25) -357.292
('E', 0, 0, 'R', 25) HIT ('E', 0, 0, 'R', 0) -376.975
Overall Reward: -326.975
```

Simulation Example 2:

```
START STATE: ('C', 2, 0, 'R', 100)
('C', 2, 0, 'R', 100) UP ('N', 2, 0, 'R', 100) -20.0
('N', 2, 0, 'R', 100) CRAFT ('N', 1, 1, 'R', 100) -39.98
('N', 1, 1, 'R', 100) CRAFT ('N', 0, 2, 'D', 100) -59.94
('N', 0, 2, 'D', 100) DOWN ('C', 0, 2, 'D', 100) -79.88
('C', 0, 2, 'D', 100) RIGHT ('E', 0, 2, 'D', 100) -99.8
('E', 0, 2, 'D', 100) HIT ('E', 0, 2, 'D', 50) -119.7
('E', 0, 2, 'D', 50) SHOOT ('E', 0, 1, 'D', 25) -139.581
('E', 0, 1, 'D', 25) SHOOT ('E', 0, 0, 'R', 0) -159.441
Overall Reward: -109.441
```

We find after multiple simulations that the best strategy for IJ is to go up from `C` to `N` and keep crafting arrows until MM becomes dormant. The fact that MM is in the ready state initially and IJ is in `C` state means that MM's attack on IJ can be successful. Thus, to prevent this, MM goes to `N` where he can make

arrows which will be helpful later on to kill MM. IJ `STAYS` there until MM becomes `DORMANT`, after which IJ goes down and right to `E` to attack MM. If he has gathered arrows, he uses them as its expected value of doing damage with it is more and he keeps `HIT`ing MM until it dies.

# Task 2

## Case 1 — Indiana now on the LEFT action at East Square will go to the West Square.

The policy is exactly same as before. The fact that IJ at `E` never took action left in any of the states before, so changing its outcome doesn't make a difference. In this scenario, Value Iteration took $125$ iterations to converge.

## Case 2 — The step cost of the STAY action is now zero.

The fact that step cost of `STAY` is 0 means that if IJ is at `W`, he will stay there forever since it is better to have 0 reward than to have negative reward. At `C`, IJ prefers to `SHOOT` if he has arrows and MM's health is $25$ since there is a fair chance that MM dies and IJ gets the final reward of $50$. Further at `C`, if MM's health is greater than $25$, anyway 2 actions would be needed to kill MM so he prefers to go left to `W` always. At `E`, MM chooses either to `SHOOT`, `STAY` or go left to `C` depending upon the number of arrows he has and MM's health. If MM is ready and IJ is at `N` or `S`, he prefers to `STAY to prevent getting hurt from MM in future. At `S` is MM is dormant, IJ mostly goes up to `C` . At `N` when MM is dormant, IJ mostly goes down to `C` when MM's health is $\geq 50$ while other times using the actions `STAY` or `CRAFT`. Finally, in this case, $62$ iterations were required for convergence.

## Case 3 — Change the value of gamma to 0.25

In this scenario, value iteration converges very fast, in $8$ iterations. As the discount factor is very high, IJ tends to maximize short term rewards. It often stays at `N`, `S` and `W` while sometimes taking action `CRAFT` at `N` and `GATHER` at `S`. Also, it doesn't `HIT` unless it is at `E`. At `C`, it may `SHOOT` or `HIT` to try to kill MM while also trying to move to other positions depending upon parameters like arrows and health of MM.